

# Chapter 1

# Requirements Engineering

Dr. Padmashree Desai

# Topics covered

- Functional and non-functional requirements(Section 4.1)
- Requirements engineering processes(Section 4.2)
- Requirements elicitation(Section 4.3)
- Requirements specification(Section 4.4 brief description)
- Requirements validation Section 4.5)

# Requirements engineering

- The process of establishing the **services that a customer requires from a system** and the **constraints under which it operates** and **is developed**.
- The system requirements are the descriptions of the system services and constraints that are generated during the requirements engineering process.

# What is a requirement?

- It may range from a high-level abstract statement of a service or of a system constraint to a detailed mathematical functional specification.
- This is inevitable as requirements may serve a dual function
  - May be the basis for a bid for a contract - **therefore must be open to interpretation;**
  - May be the basis for the contract itself - **therefore must be defined in detail;**
  - Both these statements may be called requirements.

Eg. An eRetail Portal with “Select to eCart – PlaceOrder – Pay”  
Whether the payment is in Indian currency only? Done using  
Debt / credit card?

# Requirements abstraction (Davis)

“If a company wishes to let a contract for a large software development project, it must define its needs in a sufficiently abstract way that a solution is not pre-defined. The requirements must be written so that several contractors can bid for the contract, offering, perhaps, different ways of meeting the client organization’s needs. Once a contract has been awarded, the contractor must write a system definition for the client in more detail so that the client understands and can validate what the software will do. Both of these documents may be called the requirements document for the system.”

# Types of requirement

- User requirements
  - Statements in natural language plus diagrams of the services the system provides and its operational constraints. Written for customers.
- System requirements
  - A structured document setting out detailed descriptions of the system's **functions, services** and operational **constraints**. Defines what should be implemented so may be part of a contract between client and contractor.

# User and system requirements

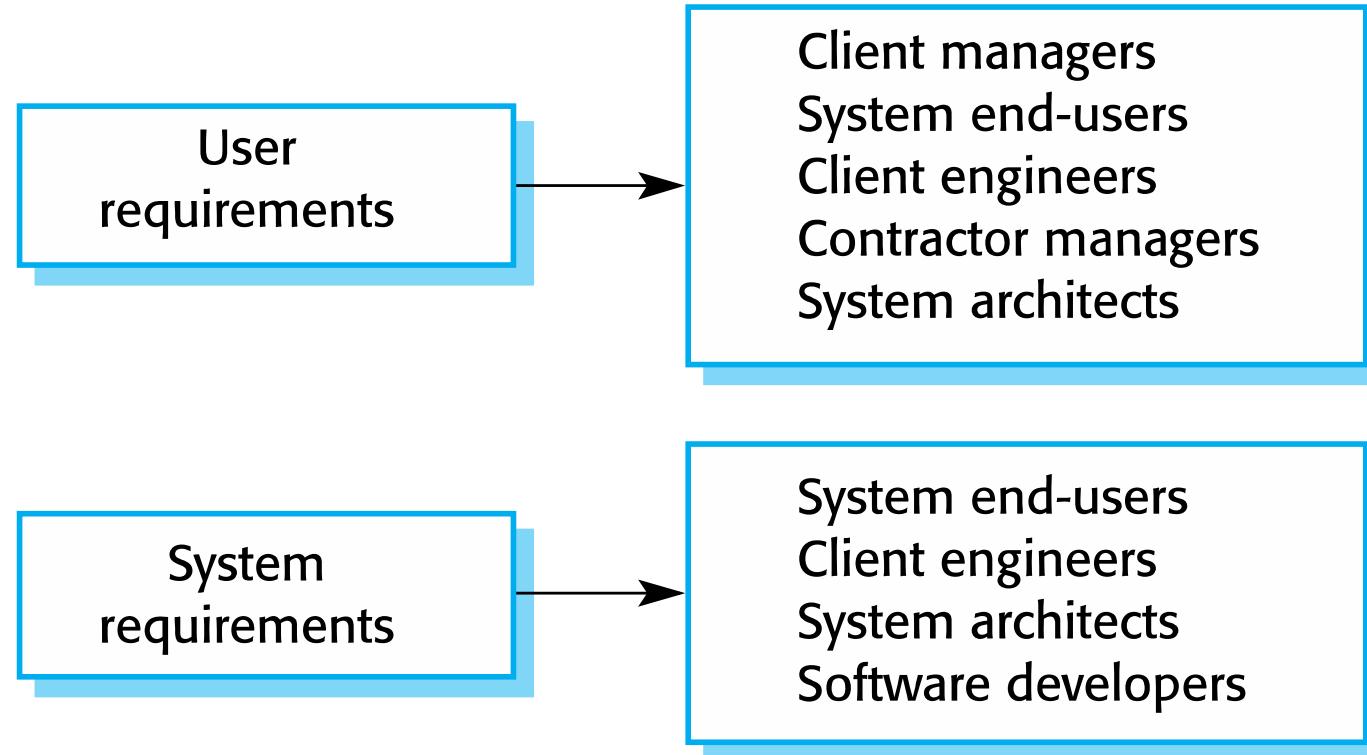
## User requirements definition

- 1.** The Mentcare system shall generate monthly management reports showing the cost of drugs prescribed by each clinic during that month.

## System requirements specification

- 1.1** On the last working day of each month, a summary of the drugs prescribed, their cost and the prescribing clinics shall be generated.
- 1.2** The system shall generate the report for printing after 17.30 on the last working day of the month.
- 1.3** A report shall be created for each clinic and shall list the individual drug names, the total number of prescriptions, the number of doses prescribed and the total cost of the prescribed drugs.
- 1.4** If drugs are available in different dose units (e.g. 10mg, 20mg, etc) separate reports shall be created for each dose unit.
- 1.5** Access to drug cost reports shall be restricted to authorized users as listed on a management access control list.

# Readers of different types of requirements specification



# System stakeholders

- Any person or organization who is affected by the system in some way and so who has a legitimate interest
- Stakeholder types
  - End users
  - System managers
  - System owners
  - External stakeholders

# Stakeholders in the Mentcare system

- Patients whose information is recorded in the system.
- Doctors who are responsible for assessing and treating patients.
- Nurses who coordinate the consultations with doctors and administer some treatments.
- Medical receptionists who manage patients' appointments.
- IT staff who are responsible for installing and maintaining the system.

# Stakeholders in the Mentcare system

- A medical ethics manager who must ensure that the system meets current ethical guidelines for patient care.
- Health care managers who obtain management information from the system.
- Medical records staff who are responsible for ensuring that system information can be maintained and preserved, and that record keeping procedures have been properly implemented.

# Agile methods and requirements

- Many agile methods argue that producing detailed system requirements is a waste of time as requirements change so quickly.
- The requirements document is therefore always out of date.
- Agile methods usually use incremental requirements engineering and may express requirements as ‘user stories’ (discussed in Chapter 3).
- This is **practical for business systems** but **problematic for systems that require pre-delivery analysis** (e.g. critical systems) or systems developed by several teams.

# Functional and non-functional requirements

# Functional and non-functional requirements

- Functional requirements
  - Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.
  - May state what the system should not do.
- Non-functional requirements
  - Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.
  - Often apply to the system as a whole rather than individual features or services.
- Domain requirements
  - Constraints on the system from the domain of operation

# Functional requirements

- Describe functionality or system services.
- Depend on the type of software, expected users and the type of system where the software is used.
- Functional user requirements may be high-level statements of what the system should do.
- Functional system requirements should describe the system services in detail.

# Mentcare system: functional requirements

- A user shall be able to search the appointments lists for all clinics.
- The system shall generate each day, for each clinic, a list of patients who are expected to attend appointments that day.
- Each staff member using the system shall be uniquely identified by his or her 8-digit employee number.

# Requirements imprecision

- Problems arise when functional requirements are not precisely stated.
- Ambiguous requirements may be interpreted in different ways by developers and users.
- Consider the term ‘search’ in requirement 1
  - User intention – search for a patient name across all appointments in all clinics;
  - Developer interpretation – search for a patient name in an individual clinic. User chooses clinic then search.

# Requirements completeness and consistency

- In principle, requirements should be both complete and consistent.
- Complete
  - They should include descriptions of all facilities required.
- Consistent
  - There should be no conflicts or contradictions in the descriptions of the system facilities.
- In practice, because of system and environmental complexity, it is impossible to produce a complete and consistent requirements document.

# Activity:

- Type: Pair Time: 10 mins
- Write five functional requirements of a online store selling organic products

# Solution : Functional Requirements: online store selling organic products

1. Authenticated users shall login in to a system
2. User shall be able to search a product of his interest.
3. User shall be able to add a item in a cart
4. User shall be able to delete an item from the cart
5. User shall be able to place an order cancel the order
6. User shall be able to make a payment through credit/debit/UPI
7. ----- etc

# Some Examples for Functional Requirements

1. When the user enters the information, the system shall send an approval request.
2. The server shall log all changes to existing data.
3. The system shall only allow managers to view customer banking data.
4. User shall be able retrieve his loan account details
5. System shall send notification on maturity of fixed deposits

# Non-functional requirements

- These define system properties and constraints e.g. reliability, response time and storage requirements. Constraints are I/O device capability, system representations, etc.
- Process requirements may also be specified mandating a particular IDE, programming language or development method.
- Non-functional requirements may be more critical than functional requirements. If these are not met, the system may be useless.

# Non-functional requirements implementation

- Non-functional requirements may affect the overall architecture of a system rather than the individual components.
  - For example, to ensure that performance requirements are met, you may have to organize the system to minimize communications between components.
- A single non-functional requirement, such as a security requirement, may generate a number of related functional requirements that define system services that are required.
  - It may also generate requirements that restrict existing requirements.

# Non-functional requirements implementation

- Non-functional requirements may affect the overall architecture of a system rather than the individual components.
  - For example, to ensure that performance requirements are met, you may have to organize the system to minimize communications between components.
- A single non-functional requirement, such as a security requirement, may generate a number of related functional requirements that define system services that are required.
  - It may also generate requirements that restrict existing requirements.

# Some Examples of Non-functional Requirements:

1. When 24 hours have passed since the last database backup, the server shall automatically back up the database.
2. The local terminal shall automatically print a list of orders every 4 hours.
3. When the local time is 0800, the server shall email a report of open issues to all managers.

# Contd : examples of Non Functional Requirements

1. Database security should meet HIPAA(Health Insurance Portability and Accountability Act) requirements.
2. The layout should allow users to reach their profile data from any page within 3 clicks.
3. If a user has not changed their password for 56 days, then the system should require a password change upon login.
4. The system must accommodate a minimum of 3 million concurrent users.
5. All web pages should load within 4 seconds.

# Contd: examples of Non Functional Requirements

1. The server room should accommodate a future doubling of installed hardware.
2. The server room should be accessible by authorized employees 24 hours per day.
3. The background colour for all screens must be #fff4b6

**A functional requirement frequently has a related non-functional requirement.**

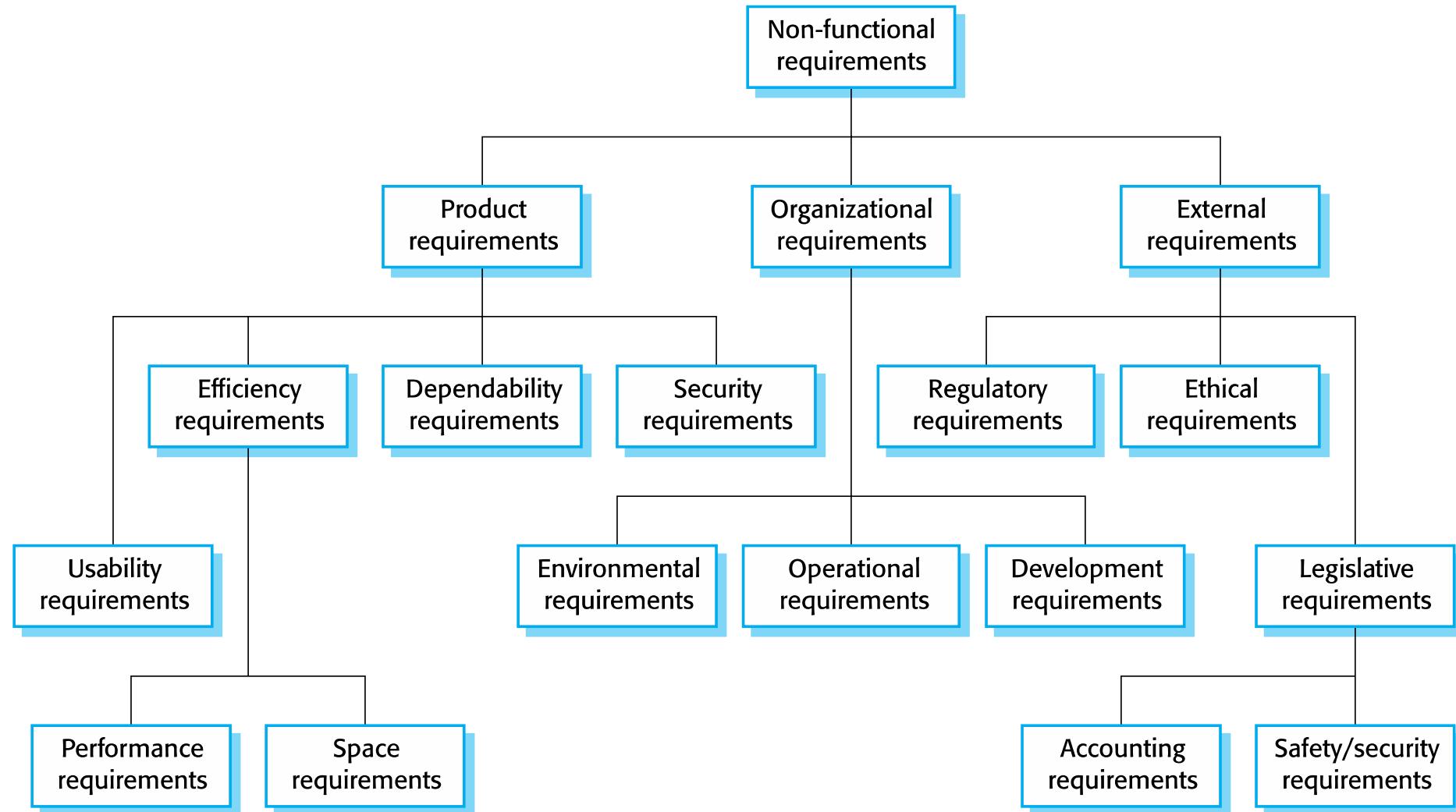
Remember that the difference is essentially the **what** and the **how**, which are both necessary.

Here are some examples:

# Functional and Related Non-functional Requirement

Functional Requirements	Related Non-Functional Requirement
When a site visitor creates an account, the server shall send a welcome email.	When sending welcome emails, the server must send them within 10 minutes of registration.
When order status changes to fulfillment, the local printer shall print a packing slip.	When packing slips are printed, they must be on both sides of 5" x 8" sheets of white paper.
The system must allow the user to fill out and submit a service form.	When the form is requested from the server, it must load with 1 second. When the submit button is pressed, it must complete upload within 2 seconds.

# Types of nonfunctional requirement



# Non-functional classifications

- Product requirements
  - Requirements which specify that the delivered product must behave in a particular way e.g. execution speed, reliability, etc.
- Organisational requirements
  - Requirements which are a consequence of organisational policies and procedures e.g. process standards used, implementation requirements, etc.
- External requirements
  - Requirements which arise from factors which are external to the system and its development process e.g. interoperability requirements, legislative requirements, etc.

# Examples of nonfunctional requirements in the Mentcare system

## **Product requirement**

The Mentcare system shall be available to all clinics during normal working hours (Mon–Fri, 0830–17.30). Downtime within normal working hours shall not exceed five seconds in any one day.

## **Organizational requirement**

Users of the Mentcare system shall authenticate themselves using their health authority identity card.

## **External requirement**

The system shall implement patient privacy provisions as set out in HStan-03-2006-priv.

# Goals and requirements

- Non-functional requirements may be very difficult to state precisely and imprecise requirements may be difficult to verify.
- Goal
  - A general intention of the user such as ease of use.
- Verifiable non-functional requirement
  - A statement using some measure that can be objectively tested.
- Goals are helpful to developers as they convey the intentions of the system users.

# Example : Usability requirements(Mentcare)

- The system should be easy to use by medical staff and should be organized in such a way that user errors are minimized. (Goal)
- Medical staff shall be able to use all the system functions after four hours of training. After this training, the average number of errors made by experienced users shall not exceed two per hour of system use. (Testable non-functional requirement)

# KLE STUDIO - Nonfunctional Requirements

- **Response time -**
  - User must be able to see the homepage of the application within 5 seconds after login about 90% of times.
  - Response time of all the options in the application must be less than 5 seconds about 90% of times for navigation.
  - The response time also depends on the speed of the network and device speed.
- **Reliability –**
  - The software application that we have featured accepts legitimate inputs and produces expected results under all conditions about 95% of times.
- **Availability –**
  - The data is available 24 x 7 except during maintenance.
- **Usability-**
  - The application meets usability requirements by providing well-formed graphical user interface and informative error messages.

# Hints: Non Functional Requirements

- All the back end processing , data updation and any thing required to perform functional requirement should be included in Non-functional requirements.
  - Example: For predicting sale in the current week, last 5 days of realtime data has to be updated.
  - Or Tweeter data shall be analysed periodically on trending #tags and keep the analysis results ready. These are just examples.

Valid Non requirement	Invalid Non- requirement
<ul style="list-style-type: none"><li>•The response time of command x should be less than one second 90% of the times</li><li>•A transaction should be processed in less than one second 98% of the time</li></ul>	<ul style="list-style-type: none"><li>•Response time should be good</li><li>•Process transaction quickly</li></ul>

# Metrics for specifying nonfunctional requirements

Property	Measure
Speed	Processed transactions/second User/event response time Screen refresh time
Size	Mbytes Number of ROM chips
Ease of use	Training time Number of help frames
Reliability	Mean time to failure Probability of unavailability Rate of failure occurrence Availability
Robustness	Time to restart after failure Percentage of events causing failure Probability of data corruption on failure
Portability	Percentage of target dependent statements Number of target systems

# Activity:

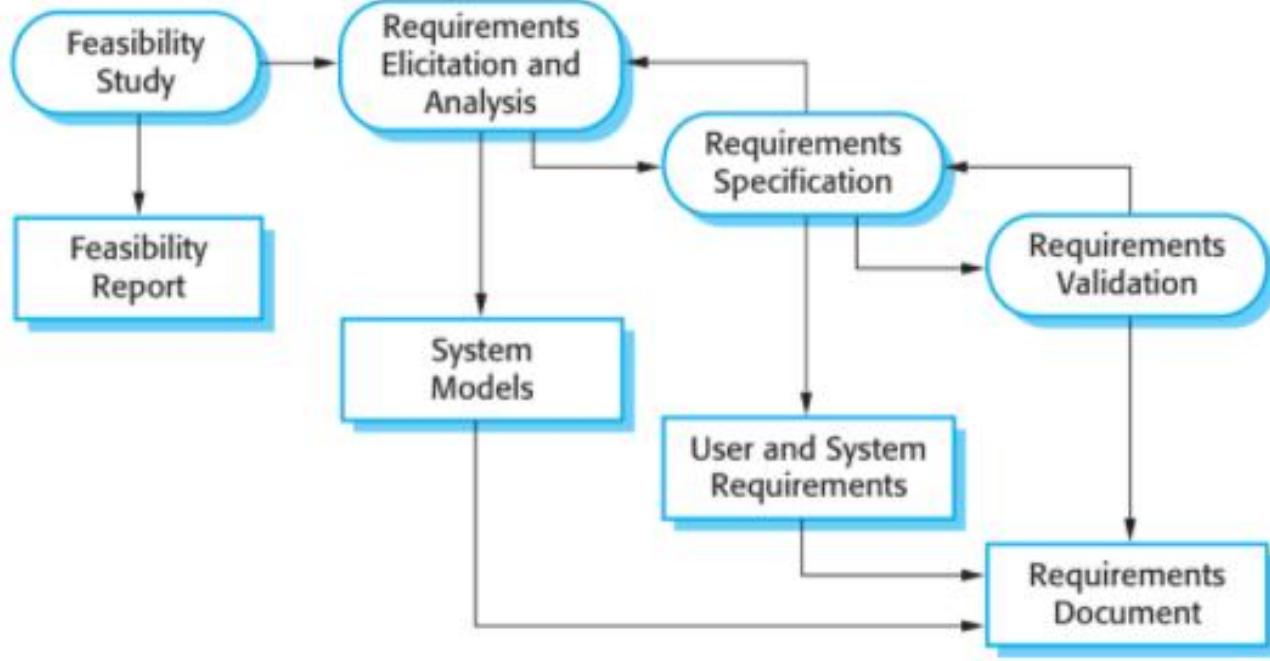
- Type: Pair Time: 10 mins
- Write non-functional requirements of a online store selling organic products for which you have already written functional requirements.

# Solution: Non-Functional Requirements for Online store selling organic products

- Alert message should be sent for the user who tries three times during login while entering password.
- OTP should be sent to personal mail-id or mobile during the payment for order
- Response time should be less than 1 sec.
- System should be portable on all devices and OS.
- Graphical User interface should provide context help while using the system
- Amount should be refunded within 1 week of working hours... etc

# Requirements engineering processes

# Requirement Engineering process

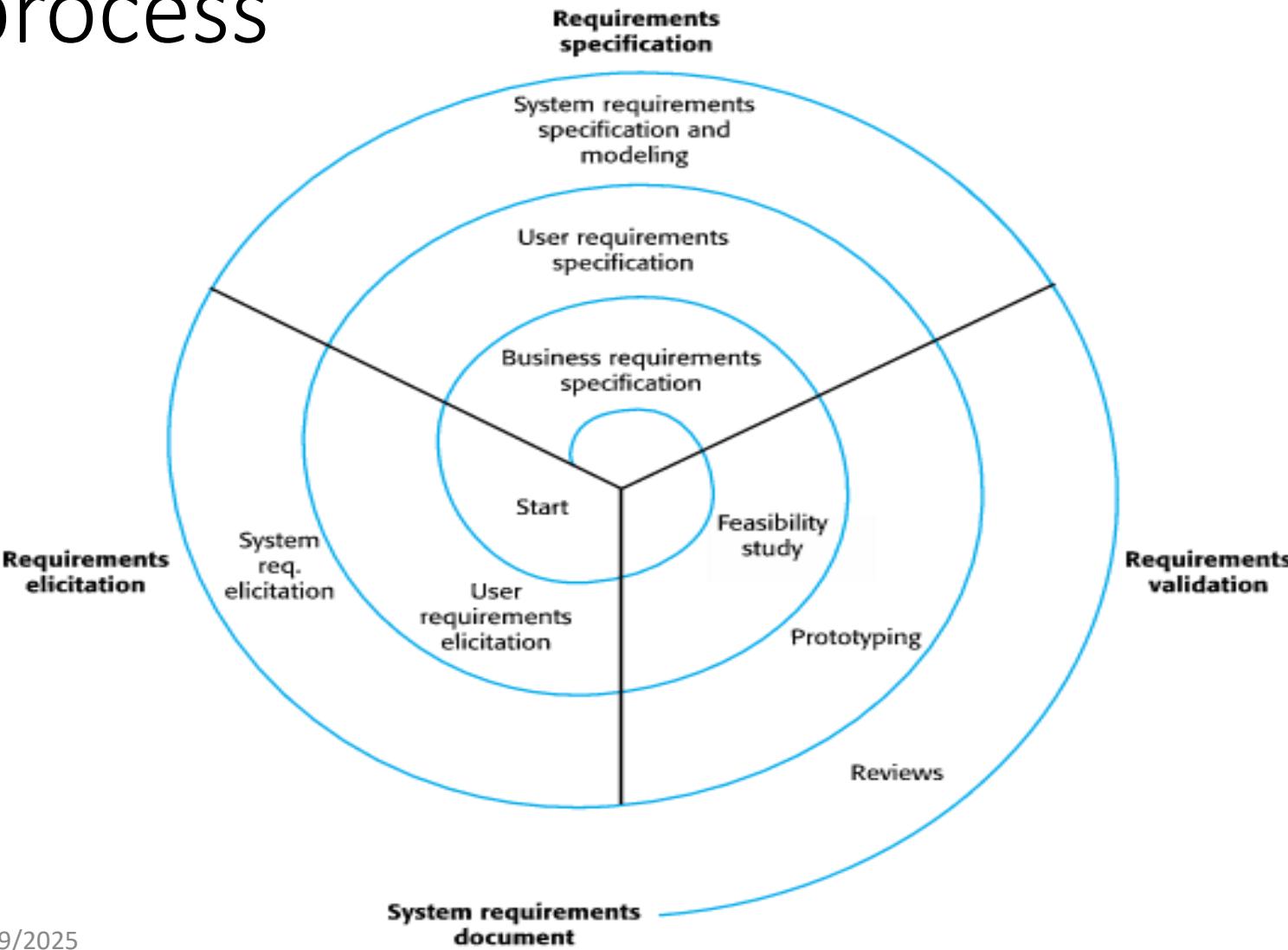


- The processes used for RE vary widely depending on the application domain, the people involved and the organisation developing the requirements.
- In practice, RE is an iterative activity in which these processes are interleaved.

# Requirements engineering processes

- The processes used for RE vary widely depending on the application domain, the people involved and the organisation developing the requirements.
- In practice, RE is an iterative activity in which these processes are interleaved.

# A spiral view of the requirements engineering process



- The activities are organized as an iterative process around a spiral. The output of the RE process is a system requirements document.
- The amount of time and effort devoted to each activity in an iteration depends on the stage of the overall process, the type of system being developed, and the budget that is available.
- The number of iterations around the spiral can vary so that the spiral can be exited after some or all of the user requirements have been elicited.
- Agile development can be used instead of prototyping so that the requirements and the system implementation are developed together.

# Requirements elicitation

# Requirements elicitation and analysis

- Sometimes called requirements elicitation or requirements discovery.
- Involves technical staff working with customers to find out about the application domain, the services that the system should provide and the system's operational constraints.
- May involve end-users, managers, engineers involved in maintenance, domain experts, trade unions, etc. These are called *stakeholders*.

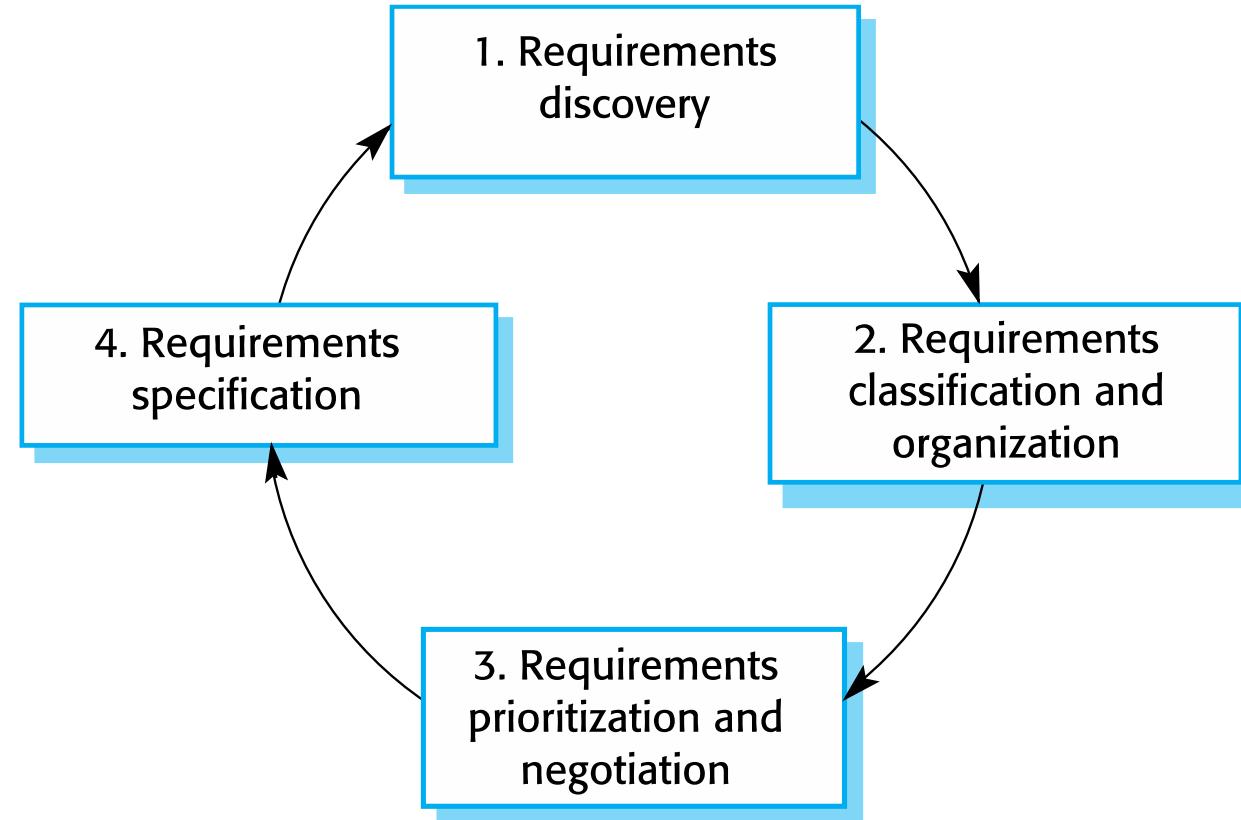
# Requirements elicitation

- Software engineers work with a range of system stakeholders to find out about the application domain, the services that the system should provide, the required system performance, hardware constraints, other systems, etc.
- Stages include:
  - Requirements discovery,
  - Requirements classification and organization,
  - Requirements prioritization and negotiation,
  - Requirements specification.

# Problems of requirements elicitation

- Stakeholders don't know what they really want.
- Stakeholders express requirements in their own terms.
- Different stakeholders may have conflicting requirements.
- Organisational and political factors may influence the system requirements.
- The requirements change during the analysis process. New stakeholders may emerge and the business environment may change.

# The requirements elicitation and analysis process



# Process activities: The requirements elicitation and analysis process

- Requirements discovery
  - Interacting with stakeholders to discover their requirements. Domain requirements are also discovered at this stage.
- Requirements classification and organisation
  - Groups related requirements and organises them into coherent clusters.
- Prioritisation and negotiation
  - Prioritising requirements and resolving requirements conflicts.
- Requirements specification
  - Requirements are documented and input into the next round of the spiral.

# Requirements discovery

- The process of gathering information about the required and existing systems and distilling the user and system requirements from this information.
- Interaction is with system stakeholders from managers to external regulators.
- Systems normally have a range of stakeholders.

## Methods

1. Interviews
2. Ethnography
3. Stories and scenarios
4. Use cases

# Interviewing

- Formal or informal interviews with stakeholders are part of most RE processes.
- Types of interview
  - Closed interviews based on pre-determined list of questions
  - Open interviews where various issues are explored with stakeholders.
- Effective interviewing
  - Be open-minded, avoid pre-conceived ideas about the requirements and are willing to listen to stakeholders.
  - Prompt the interviewee to get discussions going using a springboard question, a requirements proposal, or by working together on a prototype system.

# Interviews in practice

- Normally a mix of closed and open-ended interviewing.
- Interviews are good for getting an overall understanding of what stakeholders do and how they might interact with the system.
- Interviewers need to be open-minded without pre-conceived ideas of what the system should do
- You need to prompt the user to talk about the system by suggesting requirements rather than simply asking them what they want.

# Problems with interviews

- Application specialists may use language to describe their work that isn't easy for the requirements engineer to understand.
- Interviews are not good for understanding domain requirements
  - Requirements engineers cannot understand specific domain terminology;
  - Some domain knowledge is so familiar that people find it hard to articulate or think that it isn't worth articulating.

# Ethnography

- A social scientist spends a considerable time observing and analysing how people actually work.
- People do not have to explain or articulate their work.
- Social and organisational factors of importance may be observed.
- Ethnographic studies have shown that work is usually richer and more complex than suggested by simple system models.

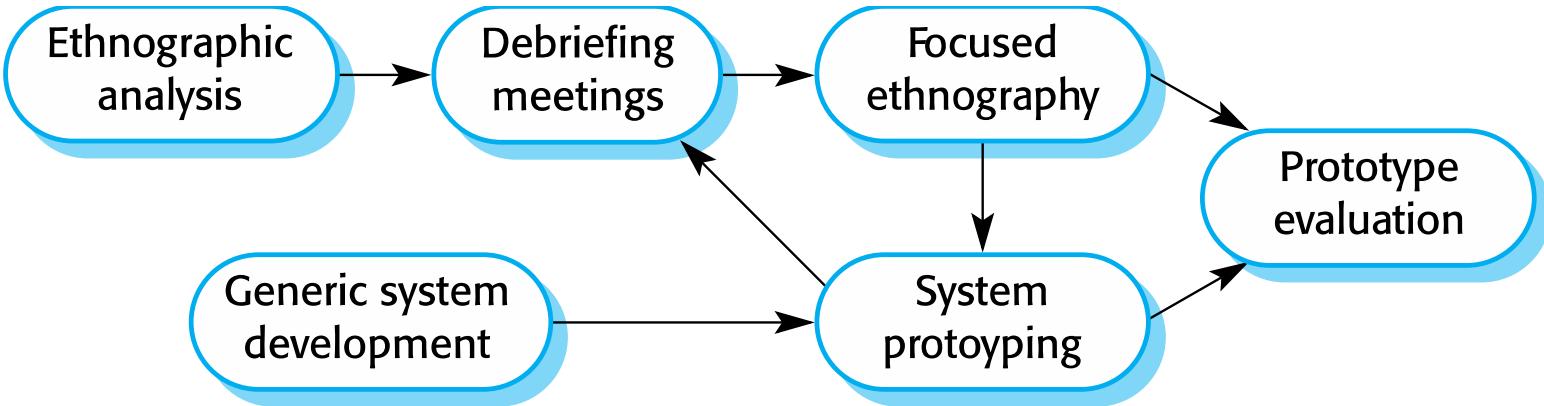
# Scope of ethnography

- Requirements that are derived from the way that people actually work rather than the way I which process definitions suggest that they ought to work.
- Requirements that are derived from cooperation and awareness of other people's activities.
  - Awareness of what other people are doing leads to changes in the ways in which we do things.
- Ethnography is effective for understanding existing processes but cannot identify new features that should be added to a system.

# Focused ethnography

- Developed in a project studying the air traffic control process
- Combines ethnography with prototyping
- Prototype development results in unanswered questions which focus the ethnographic analysis.
- The problem with ethnography is that it studies existing practices which may have some historical basis which is no longer relevant.

# Ethnography and prototyping for requirements analysis



# Stories and scenarios

- Scenarios and user stories are real-life examples of how a system can be used.
- Stories and scenarios are a description of how a system may be used for a particular task.
- Because they are based on a practical situation, stakeholders can relate to them and can comment on their situation with respect to the story.

# Photo sharing in the classroom (iLearn)

- Jack is a primary school teacher in Ullapool (a village in northern Scotland). He has decided that a class project should be focused around the fishing industry in the area, looking at the history, development and economic impact of fishing. As part of this, pupils are asked to gather and share reminiscences from relatives, use newspaper archives and collect old photographs related to fishing and fishing communities in the area. Pupils use an iLearn wiki to gather together fishing stories and SCRAN (a history resources site) to access newspaper archives and photographs. However, Jack also needs a photo sharing site as he wants pupils to take and comment on each others' photos and to upload scans of old photographs that they may have in their families.

Jack sends an email to a primary school teachers group, which he is a member of to see if anyone can recommend an appropriate system. Two teachers reply and both suggest that he uses KidsTakePics, a photo sharing site that allows teachers to check and moderate content. As KidsTakePics is not integrated with the iLearn authentication service, he sets up a teacher and a class account. He uses the iLearn setup service to add KidsTakePics to the services seen by the pupils in his class so that when they log in, they can immediately use the system to upload photos from their mobile devices and class computers.

# Scenarios

- A structured form of user story
- Scenarios should include
  - A description of the starting situation;
  - A description of the normal flow of events;
  - A description of what can go wrong;
  - Information about other concurrent activities;
  - A description of the state when the scenario finishes.

# Uploading photos ((iLearn))

- **Initial assumption:** A user or a group of users have one or more digital photographs to be uploaded to the picture sharing site. These are saved on either a tablet or laptop computer. They have successfully logged on to KidsTakePics.
- **Normal:** The user chooses upload photos and they are prompted to select the photos to be uploaded on their computer and to select the project name under which the photos will be stored. They should also be given the option of inputting keywords that should be associated with each uploaded photo. Uploaded photos are named by creating a conjunction of the user name with the filename of the photo on the local computer.
- On completion of the upload, the system automatically sends an email to the project moderator asking them to check new content and generates an on-screen message to the user that this has been done.

# Uploading photos

- **What can go wrong:**
- No moderator is associated with the selected project. An email is automatically generated to the school administrator asking them to nominate a project moderator. Users should be informed that there could be a delay in making their photos visible.
- Photos with the same name have already been uploaded by the same user. The user should be asked if they wish to re-upload the photos with the same name, rename the photos or cancel the upload. If they chose to re-upload the photos, the originals are overwritten. If they chose to rename the photos, a new name is automatically generated by adding a number to the existing file name.
- **Other activities:** The moderator may be logged on to the system and may approve photos as they are uploaded.
- **System state on completion:** User is logged on. The selected photos have been uploaded and assigned a status ‘awaiting moderation’. Photos are visible to the moderator and to the user who uploaded them.

# Requirements Specifications

- Requirements specification is the process of writing down the user and system requirements in a requirements document.
- User requirements are almost always written in natural language supplemented by appropriate diagrams and tables in the requirements document. System requirements may also be written in natural language, but other notations based on forms, graphical, or mathematical system models can also be used.

# Notations for writing Requirements Specifications

Notation	Description
Natural language sentences	The requirements are written using numbered sentences in natural language. Each sentence should express one requirement.
Structured natural language	The requirements are written in natural language on a standard form or template. Each field provides information about an aspect of the requirement.
Graphical notations	Graphical models, supplemented by text annotations, are used to define the functional requirements for the system. UML (unified modeling language) use case and sequence diagrams are commonly used.
Mathematical specifications	These notations are based on mathematical concepts such as finite-state machines or sets. Although these unambiguous specifications can reduce the ambiguity in a requirements document, most customers don't understand a formal specification. They cannot check that it represents what they want, and they are reluctant to accept it as a system contract. (I discuss this approach, in Chapter 10, which covers system dependability.)

# Requirements validation

# Requirements validation

- Concerned with demonstrating that the requirements define the system that the customer really wants.
- Requirements error costs are high so validation is very important
  - Fixing a requirements error after delivery may cost up to 100 times the cost of fixing an implementation error.

# Requirements checking

- Validity. Does the system provide the functions which best support the customer's needs?
- Consistency. Are there any requirements conflicts?
- Completeness. Are all functions required by the customer included?
- Realism. Can the requirements be implemented given available budget and technology
- Verifiability. Can the requirements be checked?

# Requirements validation techniques

- Requirements reviews
  - Systematic manual analysis of the requirements.
- Prototyping
  - Using an executable model of the system to check requirements. Covered in Chapter 2.
- Test-case generation
  - Developing tests for requirements to check testability.

# Requirements reviews

- Regular reviews should be held while the requirements definition is being formulated.
- Both client and contractor staff should be involved in reviews.
- Reviews may be formal (with completed documents) or informal. Good communications between developers, customers and users can resolve problems at an early stage.

# Review checks

- **Verifiability**
  - Is the requirement realistically testable?
- **Comprehensibility**
  - Is the requirement properly understood?
- **Traceability**
  - Is the origin of the requirement clearly stated?
- **Adaptability**
  - Can the requirement be changed without a large impact on other requirements?

# Requirements change

# Changing requirements

- The business and technical environment of the system always changes after installation.
  - New hardware may be introduced, it may be necessary to interface the system with other systems, business priorities may change (with consequent changes in the system support required), and new legislation and regulations may be introduced that the system must necessarily abide by.
- The people who pay for a system and the users of that system are rarely the same people.
  - System customers impose requirements because of organizational and budgetary constraints. These may conflict with end-user requirements and, after delivery, new features may have to be added for user support if the system is to meet its goals.

# Changing requirements

- Large systems usually have a diverse user community, with many users having different requirements and priorities that may be conflicting or contradictory.
  - The final system requirements are inevitably a compromise between them and, with experience, it is often discovered that the balance of support given to different users has to be changed.

Thank you